2021 60th IEEE Conference on Decision and Control (CDC)
December 13-15, 2021. Austin, Texas

Optimal CPU Scheduling in Data Centers via a Finite-Time Distributed Quantized Coordination Mechanism

Apostolos I. Rikos, Andreas Grammenos, Evangelia Kalyvianaki,
Christoforos N. Hadjicostis, Themistoklis Charalambous, and Karl H. Johansson

2021 60th IEEE Conference on Decision and Control (CDC) | 978-1-6654-3659-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CDC45484.2021.9683763

Abstract— **In this paper we analyze the problem of optimal task scheduling for data centers. Given the available resources and tasks, we propose a fast distributed iterative algorithm, which operates over a large scale network of nodes, and allows each of the interconnected nodes to reach agreement to an optimal solution in a finite number of time steps. More specifically, the algorithm (i) is guaranteed to converge to the exact optimal scheduling plan in a finite number of time steps and, (ii) once the goal of task scheduling is achieved, it exhibits distributed stopping capabilities (i.e., it allows the nodes to distributely determine whether they can terminate the operation of the algorithm). Furthermore, the proposed algorithm operates exclusively with quantized values (i.e., the information stored, processed and exchanged between neighboring agents is subject to deterministic uniform quantization) and relies on event-driven updates (e.g., to reduce energy consumption, communication bandwidth, network congestion, and/or processor usage). We also provide examples to illustrate the operation, performance, and potential advantages of the proposed algorithm. Finally, by using extensive empirical evaluations through simulations we show that the operation of our proposed algorithm is suitable for large scale networks such as data centers.**

I. Introduction

Modern Clouds infrastructure comprises a network of data centers, each containing thousands of server machines. Resource management in data centers is the procedure of allocating resources (e.g., CPU, memory, network bandwidth and disk space) to workloads such that their performance objectives are satisfied, given the available resources.

Resource allocation is inherently an optimization problem. However, solving it as such is challenging due to the scale and heterogeneity of the infrastructure and the dynamic nature of resource requirements of incoming and existing workloads. Centrally gathering all the required performance data from thousands of servers and running workloads, and solving the problem by a single solver is not ideal as gathered data becomes obsolete by the time the optimization is solved. For this reason, there has been recent interest towards practical distributed schedulers for solving this problem hierarchically. However, most of the proposed approaches employ heuristics that solve the problem approximately; see, e.g., [1], [2].

Recently, there has been a surge on distributed optimization, due to the wide variety of applications requiring related solutions, ranging from distributed estimation to machine learning [3], [4]. Most of the works in the literature consider distributed solutions with asymptotic convergence which assume that the messages/quantities exchanged among nodes in the network are real numbers and therefore converge within some error [5]. In several practical occasions, however, the quantities exchanged, such as scheduled tasks in CPU allocation, take discrete values. In addition, in many applications, such as in resource management in data centers, it is desirable to conclude the optimization in a finite number of steps via the exchange of quantized values, so that the exact solution is calculated and then applied.

In this paper, we focus on balancing the CPU utilization across data center servers by carefully deciding how to allocate CPU resources to workloads in a distributed fashion. We further take into consideration that the allocated resources take discrete (quantized) values. We propose a distributed algorithm that solves and terminates the optimization problem in a finite number of steps using quantized values. Even though the proposed algorithm could be adopted in a wide variety of applications, here, we discuss it within the context of resource management in Cloud infrastructures. The main contributions of the paper are the following.

- We present a distributed algorithm that solves the optimization problem in a finite number of time steps using quantized values.
- We deploy a distributed stopping mechanism in order to terminate the algorithm's operation, and hence the distributed optimization problem, in a finite number of time steps. This is the first distributed stopping mechanism for quantized average consensus algorithms.
- We provide an upper bound on the number of time steps needed for completion based on properties of primitive matrices. The completion time depends on connectivity (which is determined by the diameter of the network), rather than the size of the network.
- Simulations demonstrate that the proposed algorithm is suitable for large-scale networks, such as data centers.

Providing a distributed solution to the resource coordination problem on a strongly connected digraph has been studied in the literature (see, e.g., [5], [6]), but for real values and not in an optimization context. Our paper is a major departure from the current literature which mainly comprises distributed algorithms which operate with real values and exhibit asymptotic convergence within some error. Utilization of quantized values allows for more efficient usage of network resources, while finite time convergence allows calculation of the exact solution without any error. Our presented algorithm combines both characteristics and aims to pave the way for the use of fast bandwidth-efficient finite time algorithms which operate solely with quantized

values for solving resource allocation problems.

II. Notation and Preliminaries

The sets of real, rational, integer and natural numbers are denoted by $\mathbb{R}, \mathbb{Q}, \mathbb{Z}$ and $\mathbb{N}$, respectively. Symbols $\mathbb{Z}_{\geq 0}$ ($\mathbb{Z}_{>0}$) denote the sets of nonnegative (positive) integer numbers, while $\mathbb{Z}_{\leq 0}$ ($\mathbb{Z}_{<0}$) denote the sets of nonpositive (negative) integer numbers. For $a \in \mathbb{R}$, the floor $\lfloor a \rfloor$ denotes the greatest integer less than or equal to a while the ceiling $\lceil a \rceil$ denotes the least integer greater than or equal to a. Vectors are denoted by small letters, matrices are denoted by capital letters and the transpose of a matrix A is denoted by A^T. For a matrix $A \in \mathbb{R}^{n \times n}$, the entry at row i and column j is denoted by A_{ij}. By $\mathbf{1}$ we denote the all-ones vector and by I we denote the identity matrix (of appropriate dimensions).

Consider a network of n ($n \geq 2$) nodes communicating only with their immediate neighbors. The communication topology is captured by a directed graph (digraph) defined as $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$. In digraph $\mathcal{G}_d$, $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes, whose cardinality is denoted as $n = |\mathcal{V}| \geq 2$, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} - \{(v_j, v_j) \mid v_j \in \mathcal{V}\}$ is the set of edges (self-edges excluded) whose cardinality is denoted as $m = |\mathcal{E}|$. A directed edge from node v_i to node v_j is denoted by $m_{ji} \triangleq (v_j, v_i) \in \mathcal{E}$, and captures the fact that node v_j can receive information from node v_i (but not the other way around). We assume that the given digraph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$ is *strongly connected*. This means that for each pair of nodes $v_j, v_i \in \mathcal{V}$, $v_j \neq v_i$, there exists a directed *path*[1] from v_i to v_j. The diameter D of a digraph is the longest shortest path between any two nodes $v_j, v_i \in \mathcal{V}$ in the network. The subset of nodes that can directly transmit information to node v_j is called the set of in-neighbors of v_j and is represented by $\mathcal{N}_j^- = \{v_i \in \mathcal{V} \mid (v_j, v_i) \in \mathcal{E}\}$. The cardinality of $\mathcal{N}_j^-$ is called the *in-degree* of v_j and is denoted by $\mathcal{D}_j^-$. The subset of nodes that can directly receive information from node v_j is called the set of out-neighbors of v_j and is represented by $\mathcal{N}_j^+ = \{v_l \in \mathcal{V} \mid (v_l, v_j) \in \mathcal{E}\}$. The cardinality of $\mathcal{N}_j^+$ is called the *out-degree* of v_j and is denoted by $\mathcal{D}_j^+$.

Data Center and Workload Modelling

We model a data center as a set $\mathcal{V}$ of server compute nodes, each denoted by $v_i \in \mathcal{V}$, which also operate as resource schedulers; this is standard practice in modern data centers. All participating schedulers are usually interconnected with undirected communication links and, thus, the network topology forms a connected undirected graph. Nevertheless, our results are suitable for digraphs as well and, for this reason, hereafter we consider digraphs.

A job is defined as a group of tasks, and $\mathcal{J}$ denotes the set of all jobs to be scheduled. Each job $b_j \in \mathcal{J}$, $j \in \{1, \ldots, |\mathcal{J}|\}$, requires ρ_j cycles to be executed. The estimated amount of resources (i.e., CPU cycles) needed for each job is assumed to be known before the optimization starts. A job task could require resources ranging from 1 to ρ_j cycles, and the total sum of resources for all tasks of the same job is equal to ρ_j cycles. The total workload due to the jobs arriving at node v_i is denoted by l_i. The time horizon T_h is defined as the time period for which the optimization is considering the jobs to be running on the server nodes, before the next optimization decides the next allocation of resources. Hence, in this setting, the CPU capacity of each node, considered during the optimization, is computed as $\pi_i^{\max} := c_i T_h$, where c_i is the sum of all clock rate frequencies of all processing cores of node v_i given in cycles/second. The CPU availability for node v_i at optimization step m (i.e., at time mT_h) is given by $\pi_i^{\text{avail}}[m] := \pi_i^{\max} - u_i[m]$, where $u_i[k]$ is the number of unavailable/occupied cycles due to predicted or known utilization from already running tasks on the server over the time horizon T_h at step m.

Assumption 1. *We assume that the time horizon is chosen such that the total amount of resources demanded at a specific optimization step m, denoted by $\rho[m] := \sum_{b_j[m] \in \mathcal{J}[m]} \rho_j[m]$, is smaller than the total capacity of the network available, given by $\pi^{\text{avail}}[m] := \sum_{v_i \in \mathcal{V}} \pi_i^{\text{avail}}[m]$, i.e., $\rho[m] \leq \pi^{\text{avail}}[m]$.*

This assumption indicates that there is no more demand than the available resources. This assumption is realistic, since the time horizon T_h can be chosen appropriately to fulfill the requirement. In case this assumption is violated, the solution will be that all resources are being used and some workloads will not be scheduled, due to lack of resources, but how to handle this is out of the scope of this paper.

III. Problem Formulation

Consider a network $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$. Each one of the $n = |\mathcal{V}|$ nodes is endowed with a scalar quadratic local cost function $f_i : \mathbb{R}^n \mapsto \mathbb{R}$. In most cases [4], [7] a quadratic cost function of the following form is considered:

$$f_i(z) = \frac{1}{2}\alpha_i (z - \rho_i)^2, \tag{1}$$

where $\alpha_i > 0$, $\rho_i \in \mathbb{R}$ is the demand in node v_i (and in our case is a positive real number) and z is a global optimization parameter that will determine the workload at each node.

The global cost function is the sum of the local cost functions $f_i : \mathbb{R}^n \mapsto \mathbb{R}$ (shown in (1)) of every node $v_i \in \mathcal{V}$. The main goal of the nodes is to allocate the jobs in order to minimize the global cost function

$$z^* = \arg\min_{z \in \mathcal{Z}} \sum_{v_i \in \mathcal{V}} f_i(z), \tag{2}$$

where $\mathcal{Z}$ is the set of feasible values of parameter z. Optimization problem (2) can be solved in closed form and z^* is given by

$$z^* = \frac{\sum_{v_i \in \mathcal{V}} \alpha_i \rho_i}{\sum_{v_i \in \mathcal{V}} \alpha_i}. \tag{3}$$

Note that if $\alpha_i = 1$ for all $v_i \in \mathcal{V}$, the solution is the average.

Nodes require to calculate the optimal solution at every optimization step m via a distributed coordination algorithm

[1] A directed *path* from v_i to v_j exists if we can find a sequence of nodes $v_i \equiv v_{l_0}, v_{l_1}, \ldots, v_{l_t} \equiv v_j$ such that $(v_{l_{\tau+1}}, v_{l_\tau}) \in \mathcal{E}$ for $\tau = 0, 1, \ldots, t-1$.

which relies on the exchange of quantized values and converges after a finite number of time steps. The proposed algorithm allows all nodes to balance their CPU utilization (i.e., the same percentage of capacity) during the execution of the tasks, i.e.,

$$\frac{w_i^*[m]+u_i[m]}{\pi_i^{\max}} = \frac{w_j^*[m]+u_j[m]}{\pi_j^{\max}} = \frac{\rho[m]+u_{\text{tot}}[m]}{\pi^{\max}},\ \forall v_i, v_j \in \mathcal{V}, \tag{4}$$

where $w_i^*[m]$ is the *optimal* workload to be added to server node v_i at optimization step m, $\pi^{\max} := \sum_{v_i\in\mathcal{V}} \pi_i^{\max}$ and $u_{\text{tot}}[m] = \sum_{v_i\in\mathcal{V}} u_i[m]$. For simplicity of exposition, and since we consider a single optimization step, we drop index m. To achieve the requirement set in (4), we need the solution (according to (3)) to be [8]

$$z^* = \frac{\sum_{v_i\in\mathcal{V}} \pi_i^{\max} \frac{\rho_i+u_i}{\pi_i^{\max}}}{\sum_{v_i\in\mathcal{V}} \pi_i^{\max}} = \frac{\rho+u_{\text{tot}}}{\pi^{\max}}. \tag{5}$$

Hence, we modify (1) accordingly. Then, the cost function $f_i(z)$ in (1) is given by

$$f_i(z) = \frac{1}{2}\pi_i^{\max}\left(z - \frac{\rho_i+u_i}{\pi_i^{\max}}\right)^2. \tag{6}$$

In other words, each node computes its proportion of workload and from that it is able to find the workload w_i^* to receive, i.e.,

$$w_i^* = \frac{\rho+u_{\text{tot}}}{\pi^{\max}}\pi_i^{\max} - u_i. \tag{7}$$

The solution should be found in a distributed way. Specifically, we aim at developing a distributed coordination algorithm to find the solution via the exchange of information only between neighboring nodes. The algorithm should rely on processing and transmitting of quantized information while its operation should exhibit finite time convergence.

IV. Preliminaries on Distributed Coordination

A. Quantized Average Consensus

The objective of quantized average consensus problems is the development of distributed algorithms which allow nodes to process and transmit quantized information. During their operation, each node utilizes short communication packages and eventually obtains a state q^s which is equal to the largest quantized value (but not greater) or the smallest quantized value (but not lower) of the real average q of the initial quantized states, after a finite number of time steps.

In this paper we consider that quantized values are represented by integer[2] numbers. This means that each node is able to obtain a state q^s which is equal to the ceiling $\lceil q \rceil$ or the floor $\lfloor q \rfloor$ of the real average q of the initial quantized states of the nodes, after a finite number of time steps.

Since each node processes and transmits quantized information, we adopt the algorithm in [9]. Specifically, the algorithm in [9] is preliminary for our results in this paper and during its operation, each node is able to achieve quantized average consensus after a finite number of time steps. We make the following assumption which is necessary for the operation of the algorithm in [9] as well as the operation of our proposed algorithm in this paper. More specifically, Assumption 2 below is a necessary condition for each node v_j to be able to calculate the quantized average of the initial values after a finite number of time steps.

Assumption 2. *The communication topology is modeled as a strongly connected digraph.*

The operation of the algorithm presented in [9], assumes that each node v_j in the network has an integer initial state $y_j[0] \in \mathbb{Z}$. At each time step k, each node $v_j \in \mathcal{V}$ maintains its mass variables $y_j[k] \in \mathbb{Z}$ and $z_j[k] \in \mathbb{Z}_{\geq 0}$, and its state variables $y_j^s[k] \in \mathbb{Z}$, $z_j^s[k] \in \mathbb{N}$ and $q_j^s[k] = \lceil \frac{y_j^s[k]}{z_j^s[k]} \rceil$. It updates the values of the mass variables as

$$y_j[k+1] = y_j[k] + \sum_{v_i\in\mathcal{N}_j^-} \mathbb{1}_{ji}[k] y_i[k], \tag{8a}$$

$$z_j[k+1] = z_j[k] + \sum_{v_i\in\mathcal{N}_j^-} \mathbb{1}_{ji}[k] z_i[k], \tag{8b}$$

where $\mathbb{1}_{ji}[k] = 1$, if a message is received at v_j from v_i at k (otherwise, of no message is received, $\mathbb{1}_{ji}[k] = 1$).
If the following event-triggered condition holds:
(C1): $z_j[k] > 1$,
then, node v_j updates its state variables as follows:

$$z_j^s[k+1] = z_j[k+1], \tag{9a}$$

$$y_j^s[k+1] = y_j[k+1], \tag{9b}$$

$$q_j^s[k+1] = \left\lceil \frac{y_j^s[k]}{z_j^s[k]} \right\rceil. \tag{9c}$$

Then, it splits $y_j[k]$ into $z_j[k]$ equal integer pieces (with the exception of some pieces whose value might be greater than others by one). It chooses one piece with minimum y-value and transmits it to itself, and it transmits each of the remaining $z_j[k]-1$ pieces to randomly selected out-neighbors or to itself. Finally, it receives $y_i[k]$ and $z_i[k]$ from its in-neighbors, sums them with its stored $y_j[k]$ and $z_j[k]$ values (as described in (8a), (8b)) and repeats the operation.

Definition 1. *The system is able to achieve quantized average consensus if, for every $v_j \in \mathcal{V}$, there exists $k_0 \in \mathbb{Z}_+$ so that for every $v_j \in \mathcal{V}$ we have*

$$(q_j^s[k] = \lfloor q \rfloor \ \ for \ \ k \geq k_0) \ \ or \ \ (q_j^s[k] = \lceil q \rceil \ \ for \ \ k \geq k_0), \tag{10}$$

where q is the real average of the initial states defined as:

$$q = \frac{\sum_{l=1}^n y_l[0]}{n}. \tag{11}$$

The following result from [9] provides an upper bound regarding the number of time steps required for quantized average consensus to be achieved.

[2]We assume that the state of each node is integer valued. This abstraction subsumes a class of quantization effects (e.g., uniform quantization).

Theorem 1 ([9]). *The iterations in (8) and (9) allow the set of nodes to reach quantized average consensus (i.e., state variable q_j^s of each node $v_j \in \mathcal{V}$ fulfils (10)) after a finite number of steps. Specifically, for any ε, where $0 < \varepsilon < 1$, there exists $k_0 \in \mathbb{Z}_+$, so that with probability $(1-\varepsilon)^{(y^{init}+n)}$ we have $(q_j^s[k] = \lfloor q \rfloor$ for $k \geq k_0)$ or $(q_j^s[k] = \lceil q \rceil$ for $k \geq k_0)$, for every $v_j \in \mathcal{V}$, where q fulfills (11) and*

$$y^{init} = \sum_{\{v_j \in \mathcal{V}: y_j[0] > \lceil q \rceil\}} (y_j[0] - \lceil q \rceil) + \sum_{\{v_j \in \mathcal{V}: y_j[0] < \lfloor q \rfloor\}} (\lfloor q \rfloor - y_j[0]). \quad (12)$$

B. Synchronous max/min - Consensus

The max-consensus algorithm computes the maximum value of the network in a finite number of time steps in a distributed fashion [10]. If the updates of the nodes' state variables are synchronous, then the update rule for every node $v_j \in \mathcal{V}$ is:

$$x_j[k+1] = \max_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} \{x_i[k]\}. \quad (13)$$

It has been shown (see, e.g., [11, Theorem 5.4]) that the max-consensus algorithm converges to the maximum value among all nodes' initial values (i.e., to $\max\{x_i[0]\}$) in a finite number of steps s, where $s \leq D$ (D is the diameter of the communication topology). Similar results hold for the min-consensus algorithm.

V. Quantized CPU Scheduling Algorithm

In this section we propose a distributed quantized information exchange algorithm which solves the problem described in Section III. The proposed algorithm is detailed as Algorithm 1 below. Algorithm 1 allows each node v_j to calculate the optimal required workload w_j^* shown in (7), after a finite number of time steps. For solving the problem in a distributed way we make the following two assumptions.

Assumption 3. *The diameter of the network D (or an upper bound D') is known to all server nodes $v_j \in \mathcal{V}$.*

Assumption 4. *Each server node $v_j \in \mathcal{V}$ has knowledge of an upper bound $\pi^{\max}$ regarding the total capacity of the network $\pi^{\max}$ (i.e., $\pi^{\max} \geq \pi^{\max}$, where $\pi^{\max} := \sum_{v_j \in \mathcal{V}} \pi_j^{\max}$).*

Assumption 3 is necessary for coordinating the min- and max-consensus algorithm, such that each node v_j is able to determine whether convergence has been achieved and thus the operation of our algorithm needs to be terminated.

Assumption 4 is made such that our proposed algorithm allows each node v_j to calculate the correct optimal required workload w_j^* in a finite number of time steps via exchanging quantized information with its neighbors. Specifically, each node v_j needs to know $\pi^{\max}$ (where $\pi^{\max} \geq \pi^{\max}$) in order to multiply its initial value $y_j[0]$ with $\pi^{\max}$ so that $y_j[0] > z_j[0]$ (here $z_j[0]$ is a variable used by node v_j to process the value of $y_j[0]$ as it will be seen later in the proposed algorithm). Guaranteeing that $y_j[0] > z_j[0]$ is necessary during the operation of our algorithm, so that each node v_j is able to split $y_j[k]$ into $z_j[k]$ equal integer pieces (or with maximum difference between them equal to 1) at every time step $k \in \mathbb{N}$.

Remark 1. *It is interesting to note here that Algorithm 1 is based on similar principles as the algorithm presented in [12], which executes the ratio-consensus algorithm [13] (see also [14]) along with* min− *and* max−*consensus iterations [10]. Specifically, during the operation in [13], each node maintains two real valued variables and updates them by executing two parallel iterations. Then, each node is able to calculate the real average of the initial states asymptotically as the ratio of these two variables. Furthermore, by performing* min- *and* max*-consensus [10] every D time steps, each node is able to determine during which time step k_0 its state is within ε to the state of every other node (i.e., their difference is less or equal to ε). Overall, [12] allowed the nodes in the network to calculate the real average of their initial states and then terminate their operation according to a distributed stopping criterion. Nevertheless, compared to [12], Algorithm 1 has significant differences due to its quantized nature. These differences mainly focus on (i) the underlying process for calculating the quantized average of the initial states via the exchange of quantized messages, and (ii) the distributed stopping mechanism designed explicitly for quantized information exchange algorithms. Specifically, during the operation of Algorithm 1, the underlying process for calculating the quantized average of the initial states is based on [9]. This means that each node maintains two integer valued variables and updates them by executing two parallel iterations, where it splits them into integer equal pieces (or with maximum difference equal to* 1*) and transmits them to randomly chosen out-neighbors. Then, each node calculates the quantized average of the initial states in a finite number of time steps as the ceiling of the ratio of these two variables. Furthermore, the distributed stopping mechanism is based on performing* min- *and* max*-consensus every D time steps, where the* min- *and* max*-values are initialized as the floor and the ceiling of the ratio of the two integer valued variables it maintains. The* min- *and* max*-consensus converges once the* min*-values are within* 1 *of the* max*-values (i.e., their difference is less or equal to* 1*) which means that the state of every node is within* 1 *to the state of every other node. As a result, Algorithm 1 allows the nodes to calculate the quantized average of the initial states and, by utilizing the distributed stopping mechanism, to determine whether convergence has been achieved, and, thus whether the operation can be terminated.*

Next, we show that, during the operation of Algorithm 1, each node v_j is able to (i) calculate the optimal required workload w_j^* (shown in (7)) after a finite number of time steps, and (ii) after calculating w_j^* terminate its operation. Due to space limitations, we do not provide the proof for the theorem below.

Theorem 2. *Consider a strongly connected digraph $\mathcal{G}_d =$*

Algorithm 1 Quantized CPU Scheduling Algorithm

Input: A strongly connected digraph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges. Each node $v_j \in \mathcal{V}$ has knowledge of $l_j, u_j, D, \pi^{\text{upper}}, \pi_j^{\max} \in \mathbb{Z}$.

Initialization: Each node $v_j \in \mathcal{V}$ does the following:

1) Assigns a nonzero probability b_{lj} to each of its outgoing edges m_{lj}, where $v_l \in \mathcal{N}_j^+ \cup \{v_j\}$, as follows

$$b_{lj} = \begin{cases} \frac{1}{1+\mathcal{D}_j^+}, & \text{if } l = j \text{ or } v_l \in \mathcal{N}_j^+, \\ 0, & \text{if } l \neq j \text{ and } v_l \notin \mathcal{N}_j^+. \end{cases}$$

2) Sets $y_j[0] := \pi^{\text{upper}}(l_j + u_j)$, $z_j[0] = \pi_j^{\max}$, and $\text{flag}_j = 0$.

Iteration: For $k = 1, 2, \ldots,$ each node $v_j \in \mathcal{V}$, does the following:

- **while** $\text{flag}_j = 0$ **then**
  1) **if** $k \mod D = 1$ **then** sets $M_j = \lceil y_j[k]/z_j[k] \rceil$, $m_j = \lfloor y_j[k]/z_j[k] \rfloor$;
  2) broadcasts M_j, m_j to every $v_l \in \mathcal{N}_j^+$;
  3) receives M_i, m_i from every $v_i \in \mathcal{N}_j^-$;
  4) sets $M_j = \max_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} M_i$, $m_j = \min_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} m_i$;
  5) **if** $z_j[k] > 1$, **then**
     5.1) sets $z_j^s[k] = z_j[k]$, $y_j^s[k] = y_j[k]$, $q_j^s[k] = \left\lceil \frac{y_j^s[k]}{z_j^s[k]} \right\rceil$;
     5.2) sets (i) $mas^y[k] = y_j[k]$, $mas^z[k] = z_j[k]$; (ii) $c_{lj}^y[k] = 0$, $c_{lj}^z[k] = 0$, for every $v_l \in \mathcal{N}_j^+ \cup \{v_j\}$; (iii) $\delta = \lfloor mas^y[k]/mas^z[k] \rfloor$, $mas^{rem}[k] = y_j[k] - \delta\, mas^z[k]$;
     5.3) **while** $mas^z[k] > 1$, **then**
        5.3a) chooses $v_l \in \mathcal{N}_j^+ \cup \{v_j\}$ randomly according to b_{lj};
        5.3b) sets (i) $c_{lj}^z[k] := c_{lj}^z[k] + 1$, $c_{lj}^y[k] := c_{lj}^y[k] + \delta$; (ii) $mas^z[k] := mas^z[k] - 1$, $mas^y[k] := mas^y[k] - \delta$.
        5.3c) If $mas^{rem}[k] > 1$, sets $c_{lj}^y[k] := c_{lj}^y[k] + 1$, $mas^{rem}[k] := mas^{rem}[k] - 1$;
     5.4) sets $c_{jj}^y[k] := c_{jj}^y[k] + mas^y[k]$, $c_{jj}^z[k] := c_{jj}^z[k] + mas^z[k]$;
     5.5) for every $v_l \in \mathcal{N}_j^+$, if $c_{lj}^z[k] > 0$ transmits $c_{lj}^y[k]$, $c_{lj}^z[k]$ to out-neighbor v_l;
  - **else if** $z_j[k] \leq 1$, sets $c_{jj}^y[k] = y[k]$, $c_{jj}^z[k] = z[k]$;
  6) receives $c_{ji}^y[k]$, $c_{ji}^z[k]$ from $v_i \in \mathcal{N}_j^-$ and sets

$$y_j[k+1] = c_{jj}^y[k] + \sum_{v_i \in \mathcal{N}_j^-} w_{ji}[k]\; c_{ji}^y[k], \tag{14}$$

$$z_j[k+1] = c_{jj}^z[k] + \sum_{v_i \in \mathcal{N}_j^-} w_{ji}[k]\; c_{ji}^z[k], \tag{15}$$

  where $w_{ji}[k] = 1$ if node v_j receives $c_{ji}^y[k]$, $c_{ji}^z[k]$ from $v_i \in \mathcal{N}_j^-$ at iteration k (otherwise $w_{ji}[k] = 0$);
  7) **if** $k \mod D = 0$ **then**, **if** $M_j - m_j \leq 1$ **then** sets $w_j^* = \lceil q_j^s[k](\pi_j^{\max}/\pi^{\text{upper}}) \rceil$ and $\text{flag}_j = 1$.

Output: (4) holds for every $v_j \in \mathcal{V}$.

Theorem 1: Let $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges and $y_j[0] = \pi^{\text{upper}}(l_j + u_j)$, $z_j[0] = \pi_j^{\max}$ where $l_j, u_j, \pi^{\text{upper}}, \pi_j^{\max} \in \mathbb{N}$ for every node $v_j \in \mathcal{V}$ at time step $k = 0$. Suppose that each node $v_j \in \mathcal{V}$ follows the Initialization and Iteration steps as described in Algorithm 1. For any ε, where $0 < \varepsilon < 1$, there exists $k_0 \in \mathbb{N}$, so that for each node v_j it holds

$$w_j^* = \lceil q^{\text{tasks}}(\pi_j^{\max}/\pi^{\text{upper}}) \rceil = \frac{\rho + u_{\text{tot}}}{\pi^{\max}} \pi_j^{\max} - u_j,$$

with probability $(1-\varepsilon)^{(y^{init}+n)}$ where

$$q^{\text{tasks}} = \pi^{\text{upper}} \frac{\sum_{v_j \in \mathcal{V}} (l_j + u_j)}{\sum_{v_j \in \mathcal{V}} \pi_j^{\max}}, \tag{16}$$

and

$$y^{init} = \sum_{\{v_j \in \mathcal{V}: y_j[0] > \lceil q^{\text{tasks}} \rceil\}} (y_j[0] - \lceil q^{\text{tasks}} \rceil) + \sum_{\{v_j \in \mathcal{V}: y_j[0] < \lfloor q^{\text{tasks}} \rfloor\}} (\lfloor q^{\text{tasks}} \rfloor - y_j[0]), \tag{17}$$

is the total initial state error (i.e., y^{init} is the sum of the differences between (i) the value $\lceil q^{\text{tasks}} \rceil$ and the initial state $y_j[0]$ of each node v_j that has an initial state higher than the ceiling of q^{tasks} and (ii) the value $\lfloor q^{\text{tasks}} \rfloor$ and the initial state $y_j[0]$ of each node v_j that has an initial state less than the floor of q^{tasks}).

This means that each node v_j is able to (i) calculate the optimal required workload w_j^* (shown in (7)) after a finite number of time steps k_0 with probability $(1-\varepsilon)^{(y^{init}+n)}$ and (ii) after calculating w_j^*, terminate its operation.

VI. Simulation Results

In this section, we present simulation results to illustrate the behavior of our proposed distributed algorithm. In the first part, we present a random graph of 200 nodes and show how the states of the nodes converge. In the second part, we present a more quantitative analysis over a larger set of network sizes which would be more applicable to practical deployments, such as in modern data-centers. To the best of our knowledge, this is the first work that tries to tackle the problem of converging using quantized values at that scale while also providing a thorough evaluation accompanied with strong theoretical guarantees. To foster reproducibility, code, datasets, and experiments will be made publicly available[3].

Evaluation over a Small Scale Network. Here, we present how the states of the nodes converge during the iteration. The network in this example comprised 200 nodes and was randomly generated (an edge between a pair of nodes exists with probability 0.5). This process resulted in a digraph that had a diameter equal to 2. Small digraph diameters are indicative on data-center topologies and are normally preferred due to their locality and the benefit of having few hops between each node [15]. The upper bound π^{upper} of the total capacity is 1000 and the workload l_j of each node v_j

[3] https://github.com/andylamp/federated-quantized-ratio-consensus

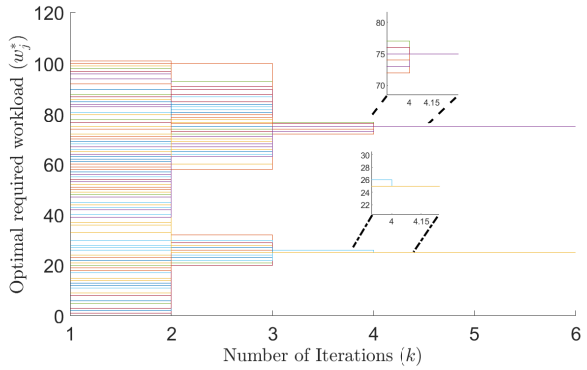

Fig. 1. Execution of Algorithm 1 over a random network comprised of 200 nodes having a diameter equal to 2. We see that the network converges in less than 10 iterations while having no oscillations.

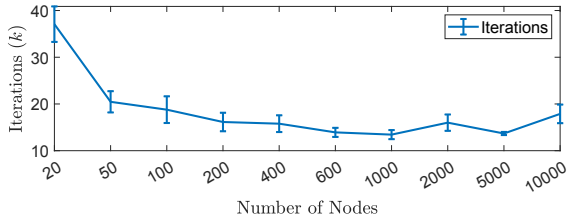

Fig. 2. Required iterations for convergence of different network sizes during the operation of Algorithm 1 along with their error bars. Each network size is evaluated across 50 trials and the aggregated values were averaged out.

was generated using a random distribution uniformly picked within the range $[1, 100]$. The node capacities $\pi_j^{\max}$ in this experiment were set to either 100 or 300 for even and odd node numbers respectively. Our simulation results are shown in Fig. 1, which depicts the load per node according to its processing capacity. We can see that the network converges monotonically within a few iterations without being affected by value oscillations or ambiguities.

Data Center Scale Evaluation. Our previous analysis dealt with a quantitative example showing the weights for all nodes involved across all iterations. Here, we present a large scale evaluation of networks over a wide gamut of sizes. Concretely, we evaluate our proposed scheme on networks sized from 20 nodes up to 10000 nodes. The topologies are randomly generated and result in digraphs that have a diameter from 2 to 10. As we previously mentioned, such digraph diameters are indicative of practical data-center deployments. We evaluated each network size across 50 trials and the aggregated values were averaged out before plotting. The upper bound of the total capacity π^{upper} for all trials was set to 1000 and the workloads were generated similarly to the previous example. We present the iterations required for all of these networks to converge; these results are shown in Fig. 2. We can see that across *all* network sizes our scheme required less than 40 iterations to converge. Another observation is that as network sizes grow, the number of iterations to converge *drops*.

VII. Conclusions and Future Directions

In this paper, we have considered the problem of optimal task scheduling for data centers. We proposed a fast distributed iterative algorithm which operates over a large scale network and allows each of the interconnected nodes to reach agreement in a finite number of time steps. In the context of task scheduling, we showed that our algorithm converges to the exact optimal scheduling plan in a finite number of time steps and then it exhibits its distributed stopping capability. Furthermore, the operation of our algorithm is event-based and relies on the exchange of quantized values between nodes in the network. Finally, we have demonstrated the performance of our algorithm shown it's fast convergence.

References

[1] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proceedings of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM, 2019, pp. 270–288.

[2] E. Boutin, J. Ekanayake, W. Lin, B. Shi, J. Zhou, Z. Qian, M. Wu, and L. Zhou, "Apollo: Scalable and coordinated scheduling for cloud-scale computing," in *Proceedings of* 11^{th} *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2014, pp. 285–300.

[3] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proceedings of IEEE*, vol. 106, no. 5, pp. 953–976, 2018.

[4] T. Yang, X. Yi, J. Wu, Y. Yuan, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Lin, and K. H. Johansson, "A survey of distributed optimization," *Annual Reviews in Control*, vol. 47, pp. 278–305, 2019.

[5] A. D. Domínguez-García and C. N. Hadjicostis, "Distributed resource coordination in networked systems described by digraphs," *Systems & Control Letters*, vol. 82, pp. 33–39, 2015.

[6] T. Charalambous, E. Kalyvianaki, C. N. Hadjicostis, and M. Johansson, "Distributed offline load balancing in MapReduce networks," in *Proceedings of* 52^{nd} *IEEE Annual Conference on Decision and Control (CDC)*, Dec 2013, pp. 835–840.

[7] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *Third International Symposium on Information Processing in Sensor Networks*, April 2004, pp. 20–27.

[8] A. Grammenos, T. Charalambous, and E. Kalyvianaki, "CPU scheduling in data centers using asynchronous finite-time distributed coordination mechanisms," *arXiv preprint arXiv:2101.06139*, 2020.

[9] A. I. Rikos, C. N. Hadjicostis, and K. H. Johansson, "Fast quantized average consensus over static and dynamic directed graphs," *arXiv preprint arXiv:2103.05172*, 2021.

[10] J. Cortés, "Distributed algorithms for reaching consensus on general functions," *Automatica*, vol. 44, pp. 726–737, March 2008.

[11] S. Giannini, D. Di Paola, A. Petitti, and A. Rizzo, "On the convergence of the max-consensus protocol with asynchronous updates," in *Proceedings of IEEE Conference on Decision and Control (CDC)*, 2013, pp. 2605–2610.

[12] S. T. Cady, A. D. Domínguez-García, and C. N. Hadjicostis, "Finite-time approximate consensus and its application to distributed frequency regulation in islanded AC microgrids," in *Proceedings of Hawaii International Conference on System Sciences*, 2015, pp. 2664–2670.

[13] A. D. Domínguez-García and C. N. Hadjicostis, "Coordination and control of distributed energy resources for provision of ancillary services," in *Proceedings of the First IEEE International Conference on Smart Grid Communications*, 2010, pp. 537–542.

[14] M. Prakash, S. Talukdar, S. Attree, V. Yadav, and M. V. Salapaka, "Distributed stopping criterion for consensus in the presence of delays," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 1, pp. 85–95, 2020.

[15] M. Besta and T. Hoefler, "Slim fly: A cost effective low-diameter network topology," in *Proceedings of the IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, 2014, pp. 348–359.